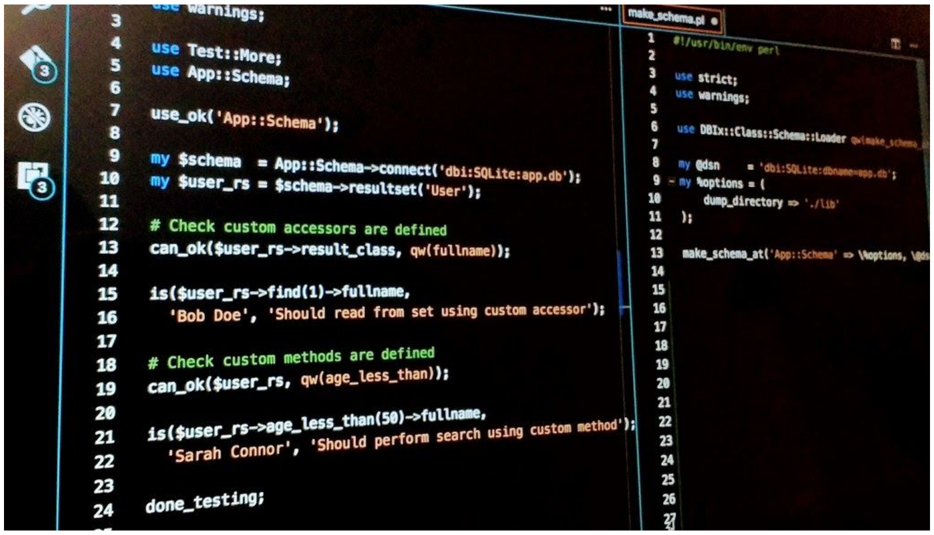
IN PREVIOUS LECTURE **(QUICK RECAP)** **Date-06/07/2020**	In today's Lecture **(Overview)**
Introduction To **Programming language**	Revision of previous Lecture
What Is a **Compiler**?? What Is **python?** What is **REPL**?? What Is **Variables**?? What is **Data Types**?? What is the Use **Concatenate**?? Commands- **Print**	Why do we learn programming on python?? Introduction To **Vscdode** input Conditional operator **if/else** **indentation** **typecasting**

Why do we learn programming on python??
=Unlike **C# and other languages**, **Python's** syntax is **human readable** and it's concise.
-As a beginner, this will allow you pick up the **basics quickly,**
-**Python** is much **popular** because it is **highly productive** as compared to other programming languages like C++ and Java
--python file Extension is **“.py”**

```
import random

# VARIABLES
GAME_OVER = False
PLAYER_HEALTH = 5
PLAYER_FOOD_POUNDS = 500
MILES_TO_GO = 2000
CURRENT_DAY = 1
CURRENT_MONTH = 3
MONTHS_31_DAYS = [3, 5, 7, 8, 10, 12]
HEALTH_DECREASES_THIS_MONTH = 0


def should_decrease_health():
    global HEALTH_DECREASES_THIS_MONTH
    if HEALTH_DECREASES_THIS_MONTH < 2:
        random_day = random.randint(0, 30)
        if random_day < 2:
            return True
    return False
```

⇒ **Introduction of Visual Studio Code**

-Visual Studio Code is a free **code editor** made by **Microsoft** for Windows



```
use warnings;

use Test::More;
use App::Schema;

use_ok('App::Schema');

my $schema  = App::Schema->connect('dbi:SQLite:app.db');
my $user_rs = $schema->resultset('User');

# Check custom accessors are defined
can_ok($user_rs->result_class, qw(fullname));

is($user_rs->find(1)->fullname,
   'Bob Doe', 'Should read from set using custom accessor');

# Check custom methods are defined
can_ok($user_rs, qw(age_less_than));

is($user_rs->age_less_than(50)->fullname,
   'Sarah Connor', 'Should perform search using custom method');

done_testing;
```

```
#!/usr/bin/env perl

use strict;
use warnings;

use DBIx::Class::Schema::Loader qw(make_schema_

my @dsn     = 'dbi:SQLite:dbname=app.db';
my %options = (
    dump_directory => './lib'
);

make_schema_at('App::Schema' => \%options, \@ds
```

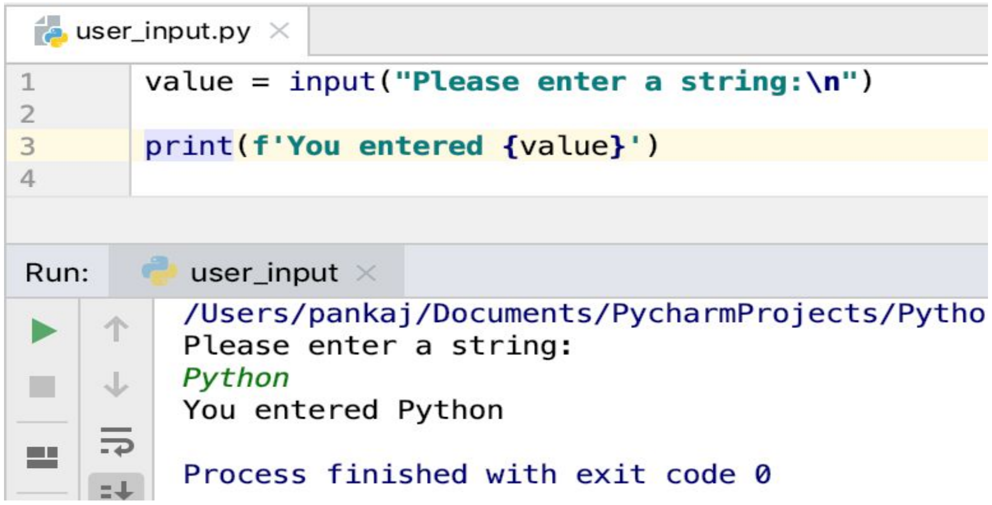
⇒ **What is IDE??**

Ide-Integrated Development Environment

⇒ **Input in python**

-The `input()` function allows user input.

-the program flow will be stopped until the user has given an **input** and has ended the **input** with the return key.



```python
value = input("Please enter a string:\n")

print(f'You entered {value}')
```

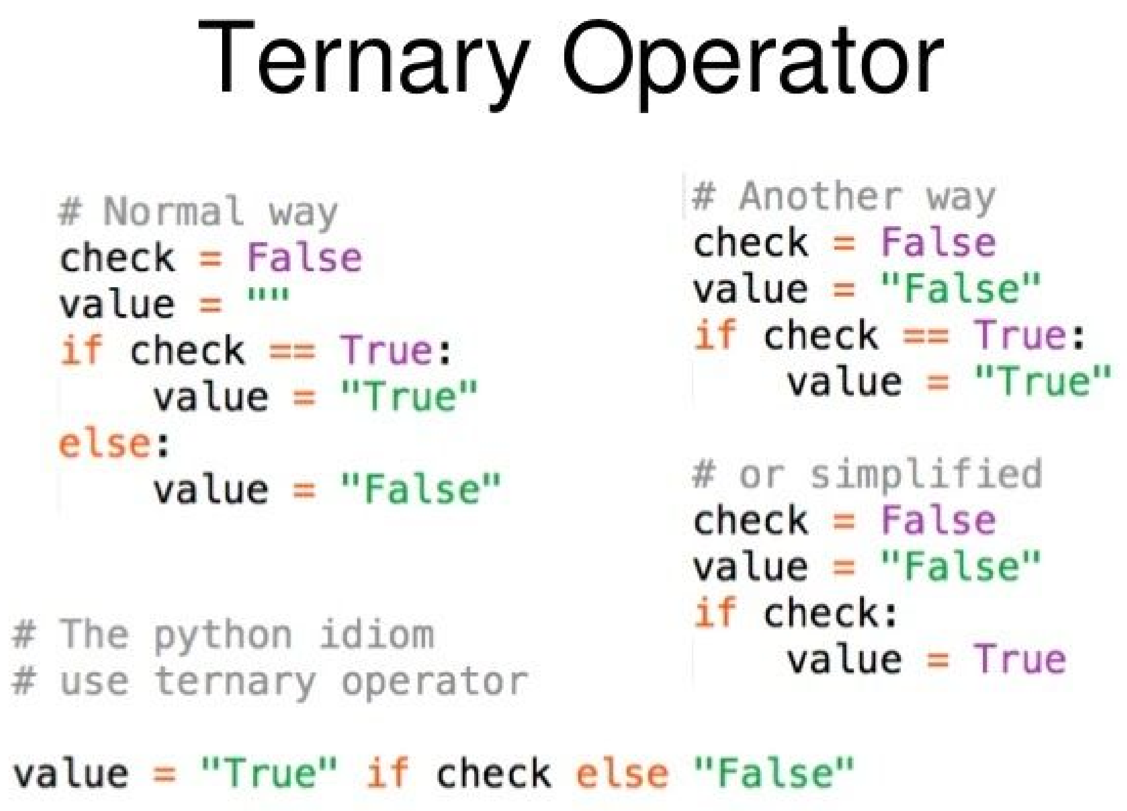
```
/Users/pankaj/Documents/PycharmProjects/Pytho
Please enter a string:
Python
You entered Python

Process finished with exit code 0
```

-To know more About it **"click here"**

⇒ **Conditional operator**

conditional expressions are **operators** that **evaluate something based on a condition being true or false.**

-it is also known as **"Ternary operator"**



Ternary Operator

```
# Normal way
check = False
value = ""
if check == True:
    value = "True"
else:
    value = "False"
```

```
# Another way
check = False
value = "False"
if check == True:
    value = "True"

# or simplified
check = False
value = "False"
if check:
    value = True
```

```
# The python idiom
# use ternary operator

value = "True" if check else "False"
```

⇒ **If/Else/elif [commands]**

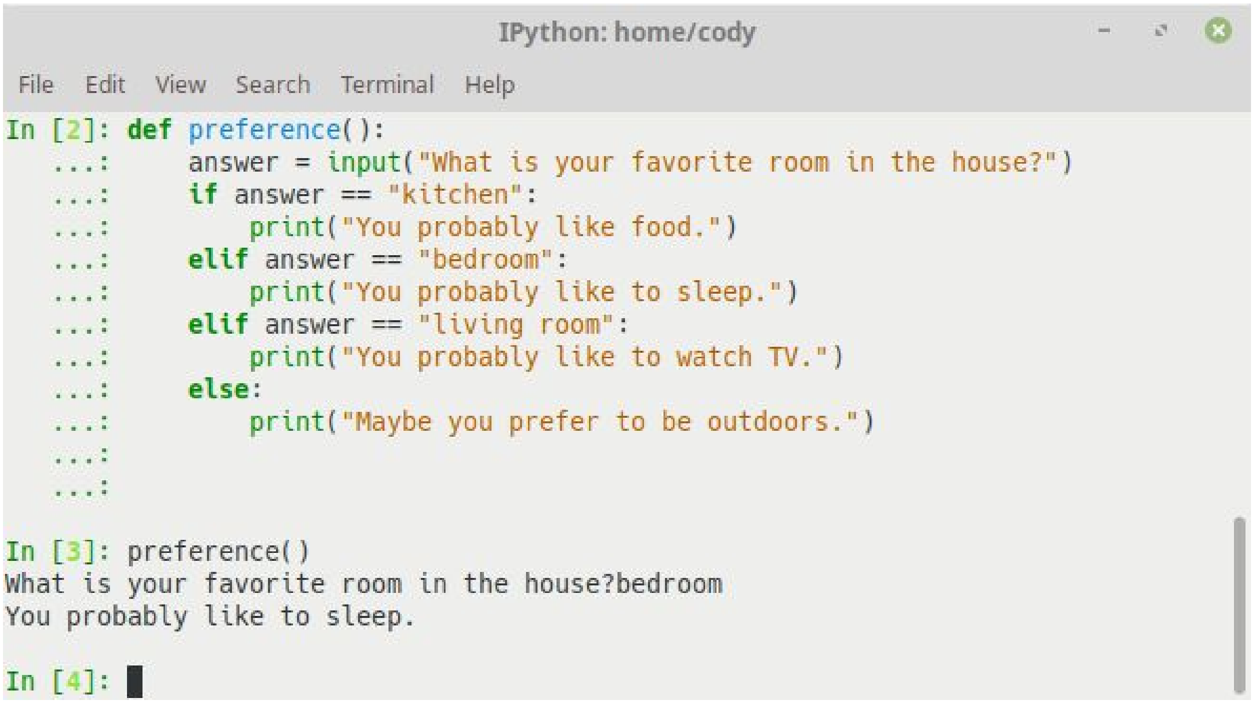
-Python supports the usual logical **conditions from mathematics**:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

These conditions can be used in several ways, most **commonly in "if statements" and loops.**

Example,

```
    a = 33
      b = 200
       if b > a:
              print("b is greater than a")
```

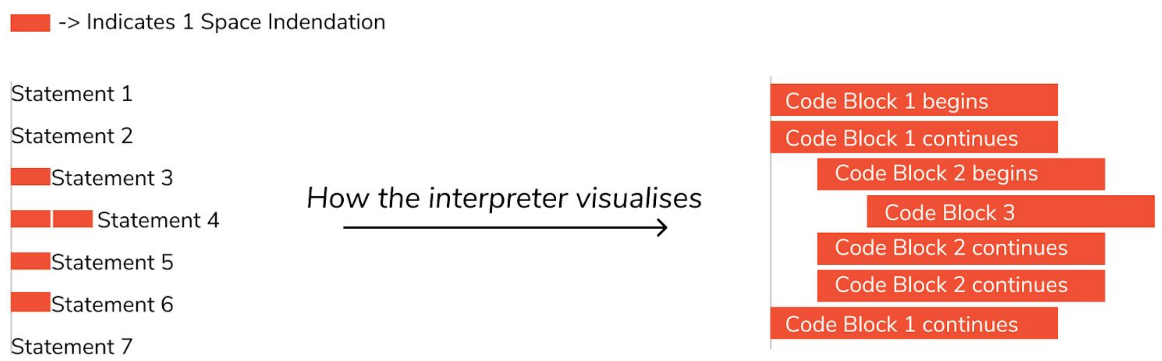


```
In [2]: def preference():
   ...:     answer = input("What is your favorite room in the house?")
   ...:     if answer == "kitchen":
   ...:         print("You probably like food.")
   ...:     elif answer == "bedroom":
   ...:         print("You probably like to sleep.")
   ...:     elif answer == "living room":
   ...:         print("You probably like to watch TV.")
   ...:     else:
   ...:         print("Maybe you prefer to be outdoors.")
   ...:
   ...:

In [3]: preference()
What is your favorite room in the house?bedroom
You probably like to sleep.

In [4]: 
```

You can also create a **simple calculator** using this command to know how to create it. "**Click Here**"

⇒ indentation

-Indentation refers to the spaces at the beginning of a code line.
-Where in other programming languages the **indentation** in code is for readability only,
-the **indentation in Python** is very important. **Python** uses **indentation** to indicate a block of code.



Here:

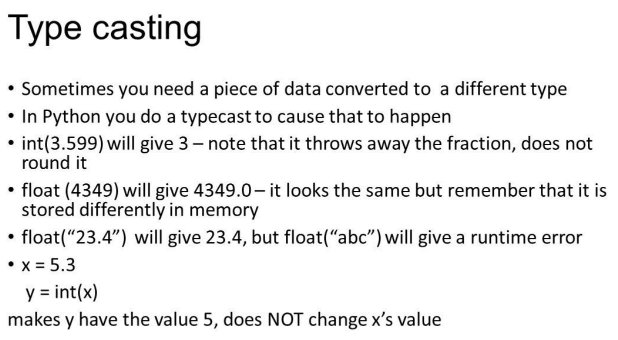
Execution happens in the same order.

Statements 1, 2, 7 belong to code block 1 as they are at the same distance to the right.
Statements 3, 5, 6 belong to code block 2
Statement 4 belongs to code block 3

⇒ **typecasting**

-there may be times when you want to specify a type on to a variable.
-This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.
-Casting in python is therefore done using constructor functions:

- **int()** - constructs an integer number from an integer literal, a float literal (by rounding down to the previous whole number), or a string literal (providing the string represents a whole number)
- **float()** - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- **str()** - constructs a string from a wide variety of data types, including strings, integer



Type casting

- Sometimes you need a piece of data converted to a different type
- In Python you do a typecast to cause that to happen
- int(3.599) will give 3 – note that it throws away the fraction, does not round it
- float (4349) will give 4349.0 – it looks the same but remember that it is stored differently in memory
- float("23.4") will give 23.4, but float("abc") will give a runtime error
- x = 5.3
  y = int(x)

makes y have the value 5, does NOT change x's value

literals and float literals